Pacific Science Review A: Natural Science and Engineering 17 (2015) 51–60

HOSTED BY

Contents lists available at ScienceDirect

Pacific Science Review A: Natural Science and Engineering

journal homepage: www.journals.elsevier.com/pacific-science-review-a-natural-science-and-engineering/

A study on rule separation based on AOP for an efficient service system

Ilwoo Choi

Kangnam University, Liberal Studies, Kangnam Univ., Giheung-gu, gyeongchun-gyan 501, Yongin, Gyeonggi-do 446-702, Republic of Korea

ARTICLE INFO

Article history:
Available online 22 March 2016

Keywords:
Aspect-oriented
Business process
Service system
Rule concern

ABSTRACT

Service-oriented architecture assures the flexibility of enterprise application development to support agile reactions when businesses change. However, the method of developing a service by combining business and constraints consumes much effort because the entire combination of logic may be changed according to the business rule changes. To improve the current method, this paper applies the aspect-oriented approach to service system development. The rule concern is proposed, and the core and cross cutting concerns of the aspect-oriented approach are included. The rule concern is extracted from business rules included in business processes and services. The rule concern is classified into process rule aspects and service rule aspects according to the level of the rule. In the proposed approach, the system is modularized into core, cross cutting and rule concerns by the separation of concerns, and they are maintained independently. Therefore, the adaptability, reusability, and maintainability of the service system will be enhanced.

Introduction

In service-oriented architecture, a service generally means that a business function is implemented into software with a collection of business rules [1]. Services are provided in heterogeneous forms by different providers and combined as a business process using business process description languages such as BPEL (business process execution language) and BPML (business process modelling language). To combine services as a business process, control flows and data flows between the services are specified, and services are orchestrated using the business process description languages [2].

According to changes in the external environment, enterprise business processes are continuously modified. Change to business processes and their corresponding IT systems are inevitable. Service-oriented architecture has several characteristics such as interoperability, location transparency and process-centric operations. These characteristics assure flexibility when developing enterprise applications, so IT systems can be constructed to agilely correspond to business changes [3].

Nevertheless, business rules that are included in the business logic of a service as the form of source code cannot typically effectively deal with the change. This approach to develop a service by combining business and constraints consumes much effort because the entire combination of logic may be changed according to the business rule changes [2,4,5].

Changes in business generally appear as business rule changes. Business rules affect the execution efficiency of the entire business process. Thus, continuous changes of business rules are time consuming because business rules must be managed, changed and implemented according to the business changes. Accordingly, various approaches such as the rule engine are used for the efficiency of service systems [5].

The aspect-oriented approach separates concerns into core and cross cutting concerns. Core concerns are extracted from the functions of the system domain, and cross cutting concerns are extracted from the common functions of several core concerns [5]. In this approach, services are designed by applying the core and cross cutting concerns of the aspect-oriented approach to improve adaptability, reusability and maintainability of service systems. Furthermore, rule concerns are proposed that the modularize system business rules. The rule concerns are divided into PRA (process rule aspect) and SRA (service rule aspect) for efficient cross cutting. The efficient modularization of business rules is possible because

E-mail address: iwchoi@kangnam.ac.kr.

Peer review under responsibility of Far Eastern Federal University, Kangnam University, Dalian University of Technology, Kokushikan University.

http://dx.doi.org/10.1016/j.psra.2015.11.005

this approach provides flexibility of service-oriented architecture and loose coupling of aspect-oriented approach. As a result, this approach provides modularity and dynamic applicability in service implementation and orchestration. These strengths are also applied to business processes, so adaptability, reusability and maintainability of the processes can also be improved [6,7].

Related works

Aspect-oriented programming

AOP (aspect-oriented programming) is introduced to address problems that cannot be solved in procedure-centric and object-oriented programming. In the procedure-centric and object-oriented programming paradigms, a system is vertically partitioned in terms of procedure or object, so functions of the system and user services are modularized. However, the horizontal requirements that are commonly applied in several modules cannot be modularized.

In the AOP paradigm, objects, components and services, which vertically present a system, are defined as core concerns, and common functions that are spread in the core concerns are defined as cross cutting concerns.

Generally, cross cutting concerns mean non-functional constraints or high level services such as log recording, security certification, transaction, resource pooling, error checking and policy application. This attempt to separate cross-cutting concerns can increase the reusability and maintainability of various objects, components and services [5].

Process-oriented composition languages

A web service is generally described as an interaction between business processes and related partners, and the interaction is described using WSDL (web service description language). The workflow processes of the BPEL (business process execution language) for web service consists of the combination activities of control flows and data flows between services [2].

Existing enterprise systems that include legacy systems generally combine business logic with business rules. The change and maintenance of these systems is time-consuming work. Process-oriented composition languages such as BPEL and BPML define the control flows and data flows between services and combine the services; however, they do not consider a method to efficiently combine continuously changing business rules with the system [4].

AO4BPEL is a concept that extends BPEL by adding an aspect-oriented concept. This attempt provides the scalability of the web service composition function of BPEL [8]. Business processes that are constructed through the service composition of BPEL can be modularized through loose coupling provided in aspects of AOP, so the reusability of the business process code can be increased. As a result, AO4BPEL provides modularity and dynamic applicability to the web service orchestration of BPEL [7].

Rule-based aspect-oriented approach

Rule concern

Services are implemented and orchestrated to deal with flow controls and service calls for executing a business process. The implementation and orchestration of services is performed by considering the reusability and scalability of a business process for actively handling the change of application, service and business process. However, the implementation and orchestration of services using existing techniques such as BPEL4WS (Business Process Execution Language for Web Services) are simply performed by composing operations and input/output information of services.

The approach in this work extends the aspect-oriented approach to actively apply business rules to services and business processes. This approach can improve adaptability, reusability and maintainability of business processes by separating business rules and functions included in the business process and service.

Fig. 1 presents the concept of concerns included in the approach. The core, cross cutting and rule concerns are defined in the approach. Core concerns present key business functions that each module should perform. Cross cutting concerns present additional

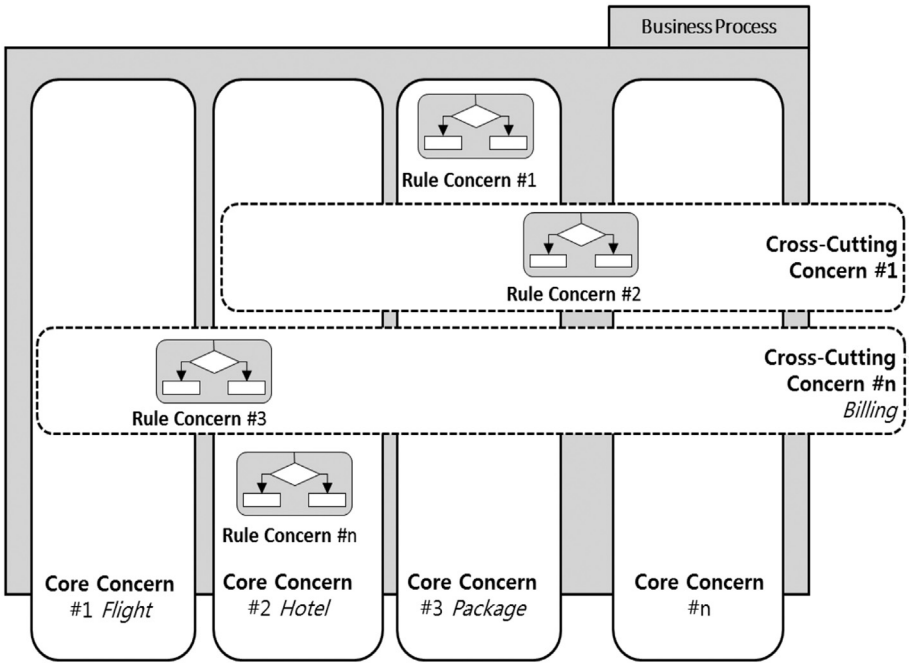


Fig. 1. Core, cross cutting, and rule concerns.

Table 1
Description of concern.

Aspect		Description
Core concern		- Core concern is the business function that each module should perform. - A business process generally consists of several services.
Cross cutting concern		- Cross cutting concern is the common system function that covers several modules. - One cross cutting concern is implemented as a service.
Rule concern	PRA (process rule aspect)	- PRA is the business rule that makes a change in the business process. - Service orchestration should be changed if PRA is changed. - PRA is generally related to the basic flow and the alternative flow of the use case description. - PRA is implemented as a part of the business process using an aspect-oriented business process execution language such as AO4BPEL or an aspect-oriented programming language such as AspectJ.
	SRA (service rule aspect)	- SRA is the business rule that is included in the service. - SRA acts as the variation point of the cross cutting concern. - SRA is generally related to the pre-condition of the use case. The pre-condition is defined using <<include>>. - SRA is implemented in a service using an aspect-oriented programming language such as AspectJ.

common functions that cover several modules. Rule concerns exist as a part of the business rules included in the core and cross cutting concerns. The rule concerns are divided into PRAs and SRAs, according to the level of rules for efficient cross cutting.

The PRA is a rule for business process orchestration in which various web services are organized into a business process. According to the PRA, a business process is changed and variation of the service orchestration occurs. The PRA is generally defined in the BPEL, so it is considered when a business process is designed.

The SRA is a rule is implemented in a service and performs a business function. The rule is included in the logic to perform or handle a function in the service. The SRA is contained in a service as codes, so it is considered when each service is designed.

Rule concerns are business rules that are separated from business functions. The rule concerns support efficient application, reuse and maintenance of service systems.

Process rule aspect

The PRA is defined as "a rule for business process orchestration that various services are organized into a related business process". Unlike a rule that is defined in a service, the PRA affects a business process. Accordingly, a variation of the business process and service orchestration occurs if a PRA is changed. PRAs are defined in BPEL statements.

A business process is generally organized by orchestrating services according to various conditions. However, the BPEL description for the business process is time-consuming work when the business process is changed in various forms. To solve the problem, this approach proposes the PRA. To efficiently deal with the frequent change of rules and services, changeable process rules are independently managed as a type of concern. It provides the efficient change and orchestration of services.

For this study, PRAs are implemented using AO4BPEL and B2J. The AO4BPEL supports the concepts of AOP such as Aspect, Joint-Point, Advice, Point-Cut and Weaving through the extension of BPEL [8,10]. The characteristics of AO4BPEL and B2J can be applicable to modularize various process rules into PRAs through the aspect of AOP and efficiently manage the rules.

Service rule aspect

In the case that services are extracted from a system domain by applying AOP, core concerns that modularize functions of the system domain and cross cutting concerns that modularize common functions between the core concerns are extracted [9]. Cross cutting concerns modularize common functions of a system domain, but in many cases, they include various rules that are specialized for the business logic of each service. Primarily, the rules included in the cross cutting concerns are described according to the service and are optionally used according to the service that calls cross cutting concerns or work as variation points which are frequently modified [7,11]. The variation points can exist in the form of codes to optionally apply proper rules for various services. In the variation

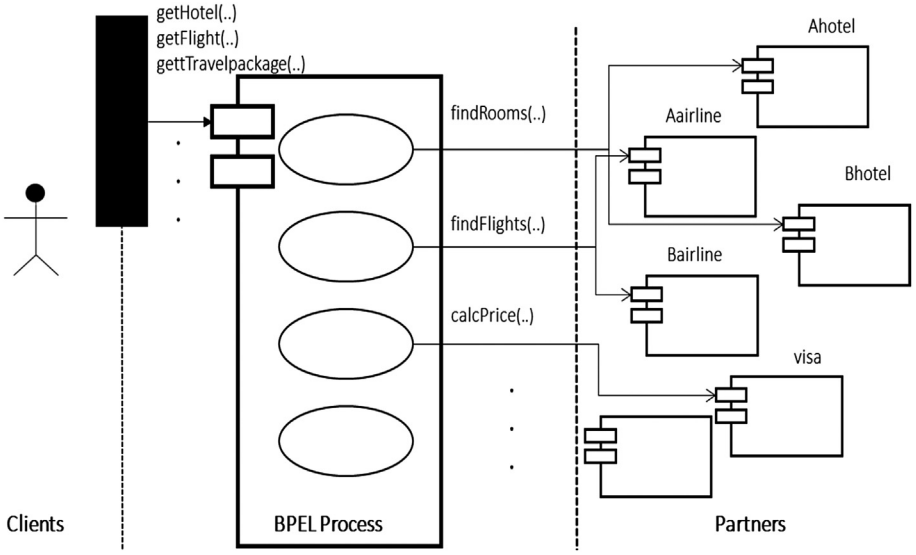


Fig. 2. Example of a travel portal as a composite web service.

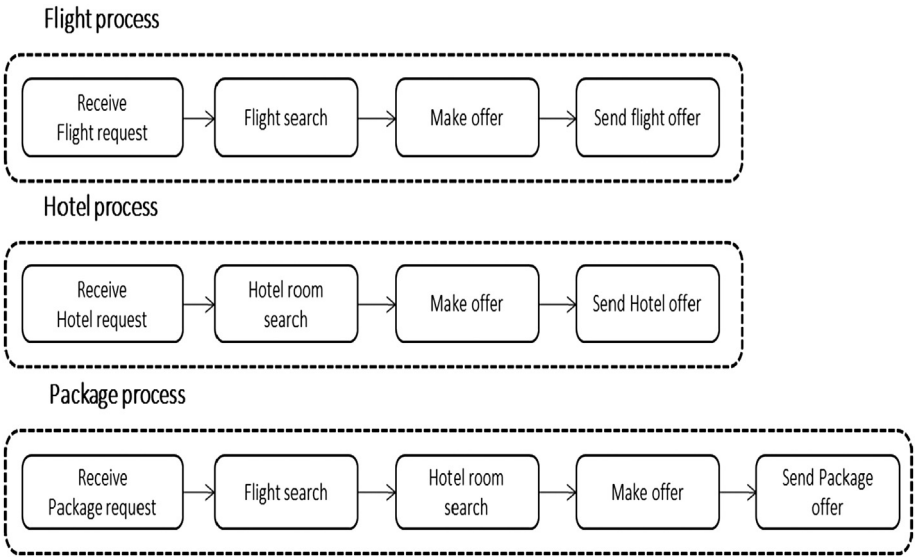


Fig. 3. Scenario of the business process for the travel portal.

points, the rules must be continuously changed according to the requirements of various services. In this approach, variation points included in the cross cutting concerns, namely variable business rules are separated into SRAs.

The SRA is defined as "a variable rule separated from business logic of each service". The extraction of the SRA is similar to the aspect separation of the AOP, but it is different from the AOP because it concerns the modularization of a variable rule included in a service. The SRA is implemented in a service using aspect-oriented languages such as AspectJ. Table 1 presents the characteristics of the proposed aspects.

Rule-based service design

Figs. 2 and 3 show an example of a travel portal web service. In the travel portal, the client requirements are achieved using the corresponding partner services through the business process orchestrated with composite services by BPEL.

Fig. 4 shows an example of a travel portal web service for describing the rule separation-based service design tasks in this research. Core and cross cutting concerns are extracted through Task 1 and Task 2, and they are modularized into services [9]. The cross cutting concerns extracted from core concerns are

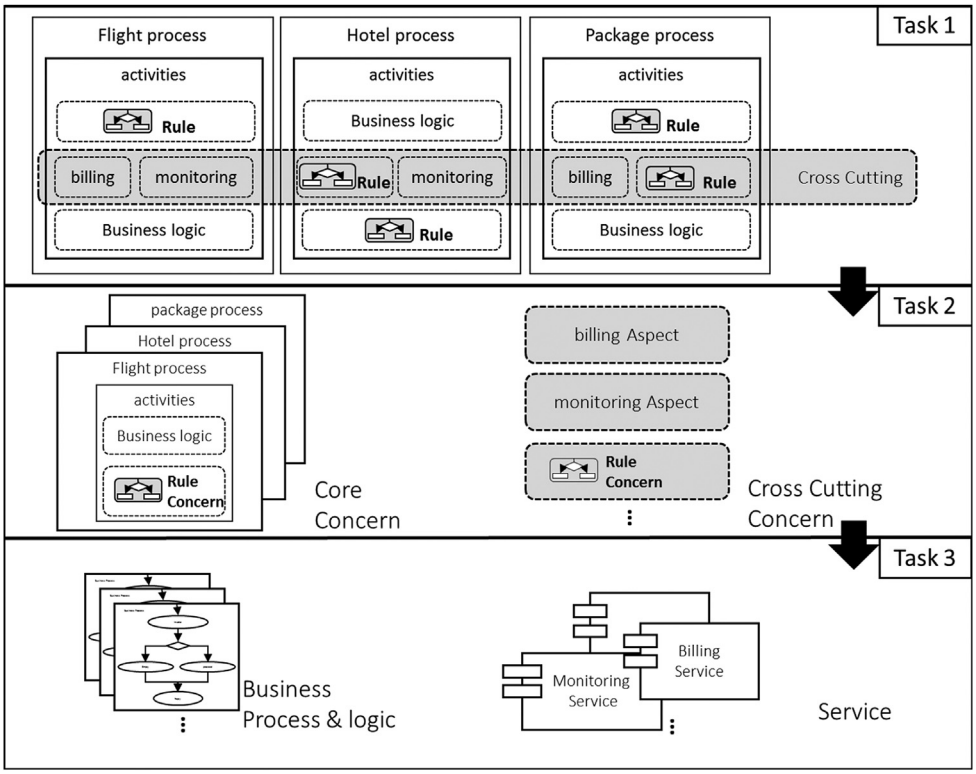


Fig. 4. Outlined tasks for applying the approach to the travel portal example.

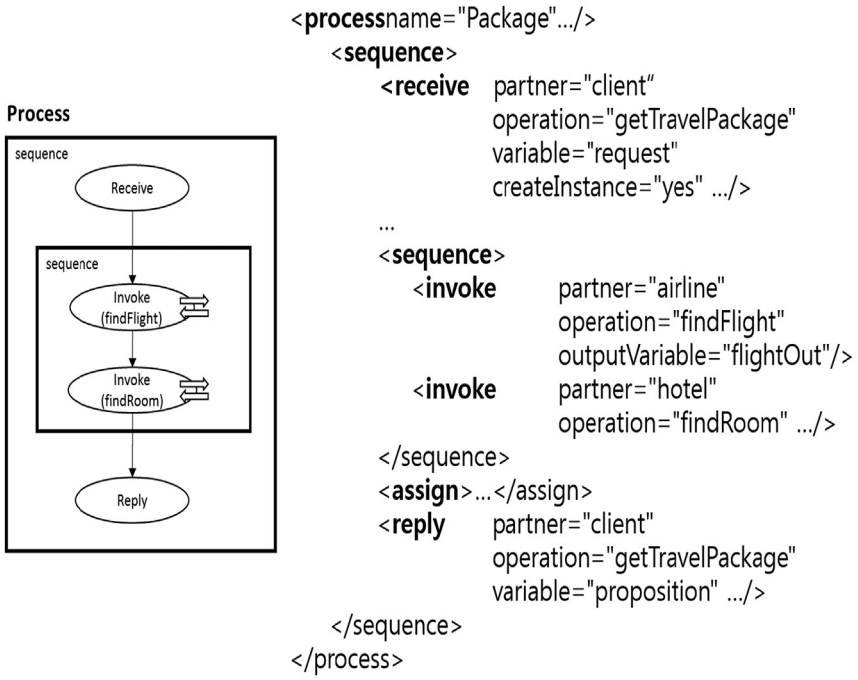


Fig. 5. Example of BPEL statements for the Package process.

modularized over maintaining loose coupling. The services are orchestrated by various business rules to compose a business process. In Task 3, the PRAs and SRAs are separated from the services, and business processes are derived from Task 1 and Task 2. The PRAs included in the core concerns and the SRAs included in the cross cutting concerns are modularized.

Process rule aspect design

Unlike a business rule that is defined in a service, a process rule affects the business process. Accordingly, a variation of service orchestration occurs if the process rule is changed. The process rule is generally defined in the BPEL statements.

Fig. 5 presents a section of a Package process written in BPEL4WS. The services provided by various partners are orchestrated through the BPEL and executed as a part of the business process. In BPEL, conditions are generally handled through <switch>, statements, and services are orchestrated according to the conditions.

If a rule such as “Do not search rooms if a flight cannot be found.” is added to the process in Fig. 5, the <switch>, statements as in Fig. 6 should be added. The rule change requires modification

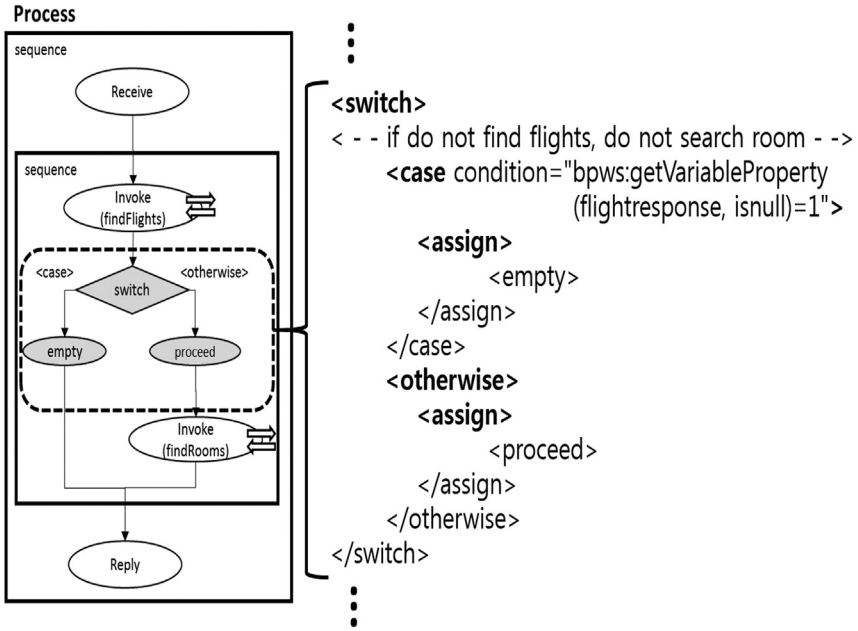
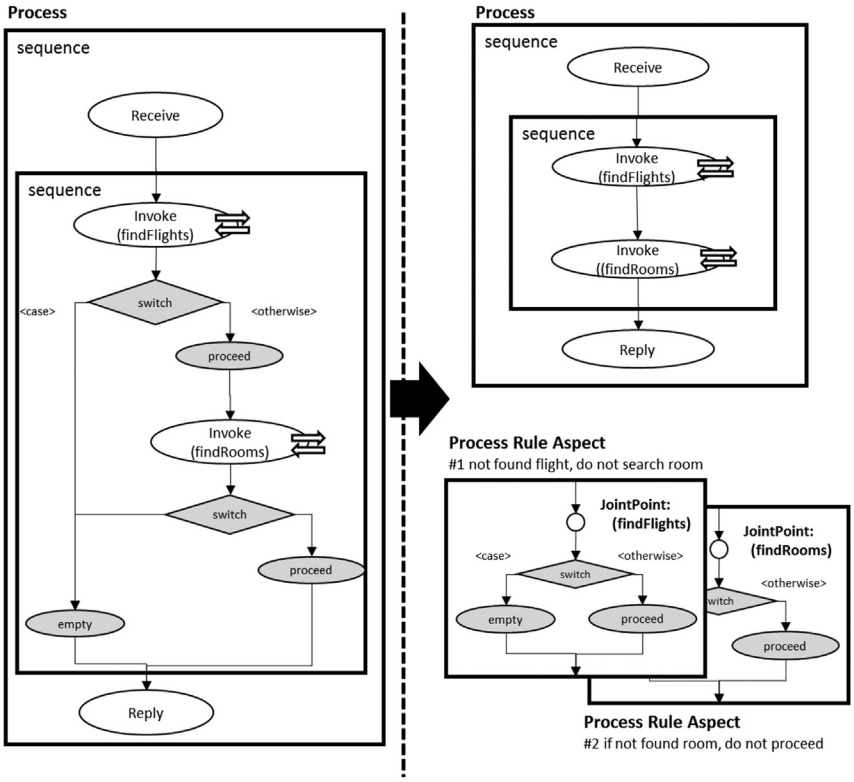


Fig. 6. Addition of BPEL statements according to the rule change.




```
// Process Rule Aspect #1
// if do not found flight, do not search room
        <aspectname="ifnotfoundflight,donotsearchroom">
        <variables>...</variables>
          <pointcutandadvicetype="around">

          <pointcutname="hotelprocurement">
           //process[@name="FullTravelPackage"]
           //invoke
           [@portType="hotelPT"and @operation="findRoom"]
          </pointcut>

          <advice>
          <switch>
          <case condition="bpws:getVariableProperty
                    (flightresponse,isnull)=1">
              <empty/>
          </case>
          <otherwise>
              <proceed>
```

Fig. 7. Example of an AO4BPEL statement for the separated Process Rule Aspects.

of the BPEL statements, so it has a continuous effect on the orchestration of a business process.

Fig. 7 presents the PRAs proposed in this approach. Each PRA defines the Joint-Point for process execution, and it is implemented using the AO4BPEL. It is implemented in the <switch>, statement within the <advice>statement, which is inserted into the Joint-Point and executed. The group of related Joint-Points is identified as a <pointcut>statement. The core and cross cutting concerns are branched from the module written for a PRA through the cross point, and the Advice is inserted into the Joint-Point and woven. Therefore, business process change is minimized because only the corresponding PRA is modified if the change of a process rule is required.

The extracted concerns can be divided into services for making a business process and the PRAs for combining the services; they can be implemented using AO4BPEL. The entire business process is not changed, and services are selectively applied according to the changes of the separated PRAs. Therefore, rule and process changes are independently performed by PRAs.

```
public class flightOutInterceptor extends AbstractInterceptor {
        public String intercept(ActionInvocation invocation) throws Exception {
        ActionContext context = invocation.getInvocationContext();

        FindFlight = (FindFlight)session.get("flight");
          if(flight==null){
                        returnAction.empty;
                        }
          returninvocation.invoke(); }}
```

Fig. 8. Example of Interceptor.

B2J is one of the service engines that support BPEL4WS. It provides various applications by transforming BPEL documents into Java source codes, differently from other BPEL engines. Currently, the BPEL does not provide intuitive conditions for analysing context information, so it is difficult to use BPEL in pervasive computing environments. To solve the problem, developers use various BPEL tools. The business rule provides high level conditions for analysing context information. Therefore, process specification and web service functions of BPEL are used as-is, and context information can be efficiently handled if a language for handling rules is added in BPEL or AO4BPEL.

AspectJ is used to add the AOP concept for efficiently managing new requirements and changeable rules to the Java source codes generated by B2J. A user writes BPEL and Interceptor documents, as in Fig. 8, and inputs the two documents into the system.

The Interceptor document has the information for inserting a routine to handle the rule of a specific part among the execution routines of BPEL. The BPEL document is transformed into Java source codes through the B2J coordinator, and the Interceptor document is transformed into AspectJ source codes through the Intercepter2AspectJ engine. The generated program source codes are compiled and transformed into executable Java class files through the AspectJ compiler and then executed in the B2J WorkerHost [10,15].

The Interceptor is an object to dynamically intercept the call of an action. The Interceptor provides a way to insert other execution codes before or after the execution of an action. Interceptor and BPEL documents are independently written by the AOP function. The Interceptor can modularize process rules as PRAs, so it is possible to easily add new requirements to BPEL documents and change requirements. The Interceptor also can more intuitively describe rules rather than AO4BPEL because it uses Java sentence structure.

Service rule aspect design

Core and cross cutting concerns extracted from a domain can be modularized into the services [9].

In Fig. 9, the Billing cross cutting concern maintains loose coupling through cross cutting and is modularized into the Billing Service; however, the function and various rules are together contained in the Billing Service as source codes like other services. The contained business rule is a variation point continuously changed according to the condition of the core concern, which calls the cross cutting concerns. This approach consumes many resources for the rule changes because the entire logic for rule composition is modified.

This problem is solved by separating the business rule of the variation point included in a service and modularizing it into an SRA, as in Fig. 10. The Billing SRA implemented through the Interceptor of AspectJ can minimize the change of the Billing Service because only the Billing SRA is modified if the rule change is required in the service.

The Interceptor provides a way to insert other execution codes before or after the execution of an action by dynamically intercepting the call of the action, so an action can be executed in a various way. The Interceptor is suitable for encapsulating the reusable common functions of several actions such as the cross cutting concerns or variation points executed in various conditions such as the SRAs [12].

Evaluation

Fig. 11 and Fig. 12 show a part of a PRA and SRA design based on a scenario of a travel portal web service domain.

The core and cross cutting concerns are extracted from the business services of the system domain through the concept of AOP.

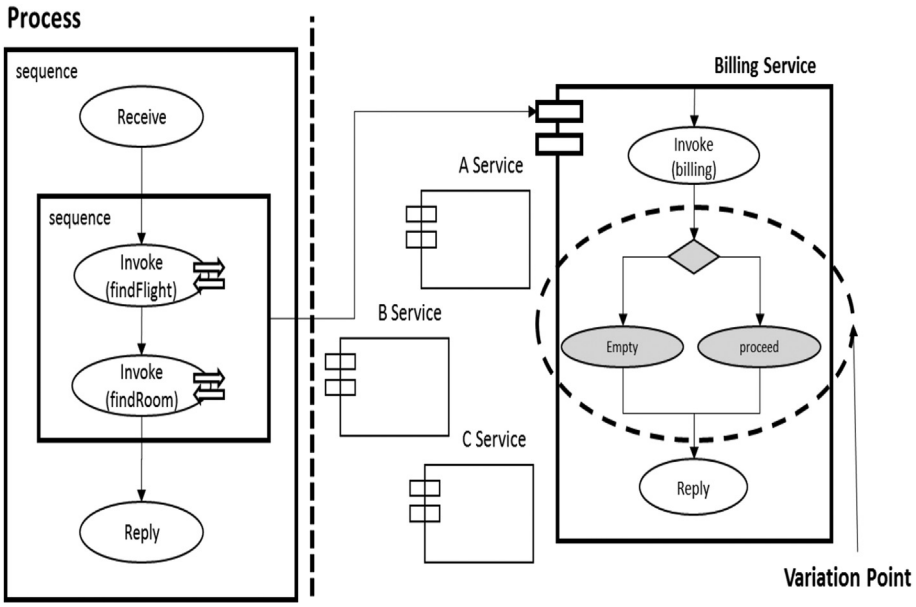


Fig. 9. Example of a variation point (Billing cross cutting concern).

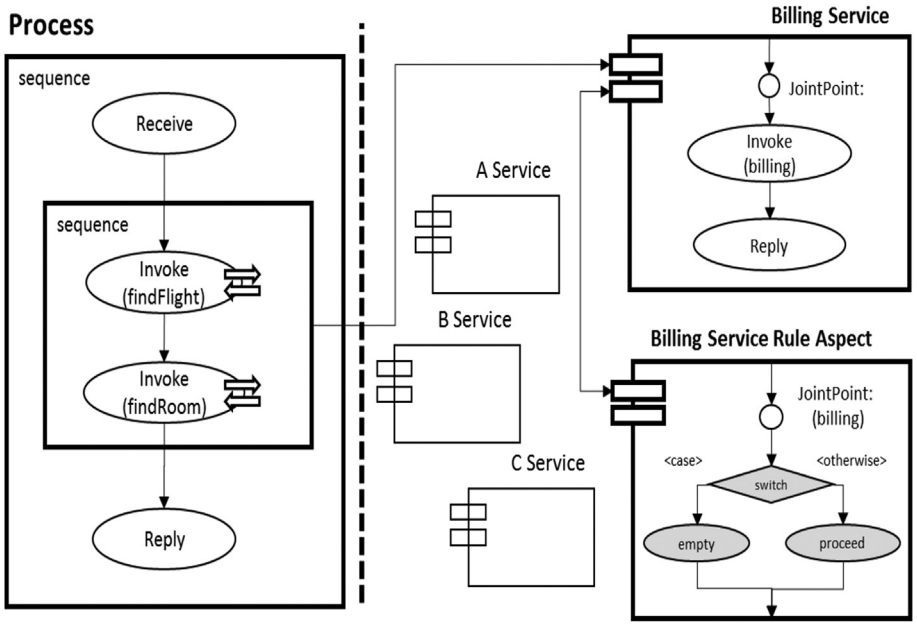


Fig. 10. Example of a service rule aspect.

The PRAs and SRAs are extracted from the core and cross cutting concerns through the concept of the proposed rule concern. The core, cross cutting and rule concerns are implemented using the aspect-oriented language and aspect-oriented business process execution languages such as AO4BPEL and AspectJ, so they are considered in the service analysis phase [13].

Table 2 presents the result of the comparison of the approach in this study with existing approaches. The separation of the rule

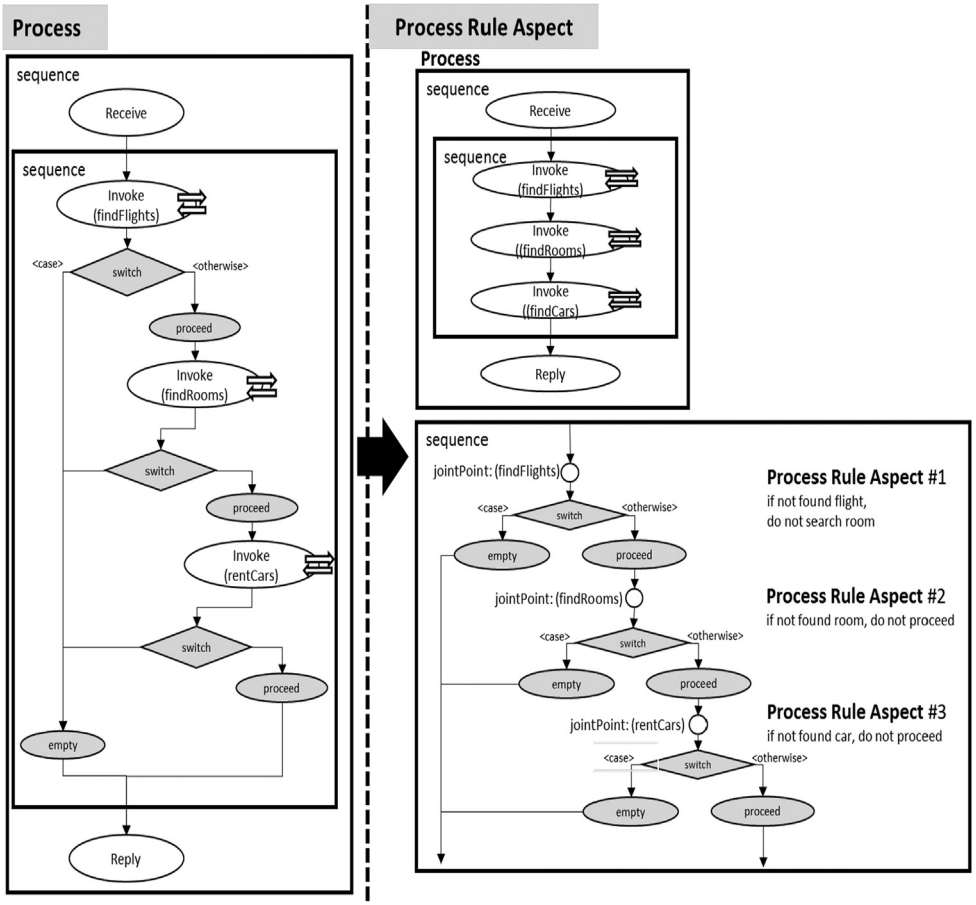


Fig. 11. Design of a process rule aspect.

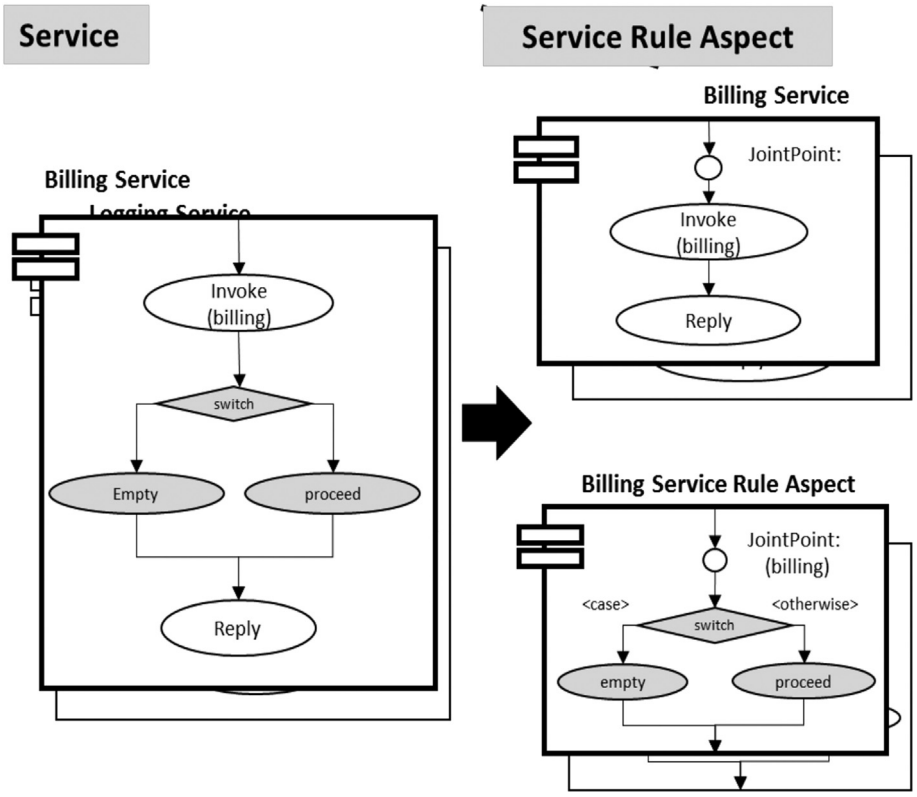


Fig. 12. Design of a services rule aspect.

Table 2
Comparison of this approach with existing approaches.

	Existing approach	This approach
Architecture	SOA	SOA
Concern	Core concern, Cross cutting concern	Core concern, Cross cutting concern, Rule concern (Process rule, Service rule)
Maintainability	Good	Higher than the existing approach
Reusability	Good	Higher than the existing approach
Adaptability	Good	Higher than the existing approach
Reusable unit	Service	Service, Process, Rule
Design complexity	Average	More complex than the existing approach

concern proposed in this approach increases the adaptability for continuous change of business rules included in business processes and services. A variation of the business process and service orchestration occurs according to business changes. In this approach, users modify only the specific rule of the corresponding PRA described with the advice of AO4BPEL or the Interceptor of AspectJ if a business policy is changed. So, the maintainability of the system is increased because the entire business process does not need modification. Additionally, modularized rules can increase the reusability of the business process codes [14].

In the view of service, this approach has an advantage in the function expansion and maintenance of the service rules because the variation points that are modularized into SRAs are modified and changed by loose coupling implemented in an aspect-oriented language such as AspectJ. However, the separation of concerns can generate an additional problem in terms of design complexity such as the complexity of the service design procedure or the scattering and entanglement of codes.

Conclusion

This paper proposed a rule-based aspect-oriented approach for efficient service system development.

The approach separates the business rules included in the business process from process functions for actively applying business rules to services and business processes. Accordingly, core, cross cutting and rule concerns are modularized by applying the concept of AOP. The rule concerns are the business rules included in a system domain.

The rule concerns are classified into process rule aspect (PRA) and service rule aspect (SRA) according to the level. The PRA is a business rule that affects business process, and it is changed for service orchestration. The SRA is a business rule that is included in unit service, and it becomes a variation point in a cross cutting concern for accepting various conditions of a core concern.

Using this approach, a service system is modularized into core, cross cutting and rule concerns through the separation of concerns, and the concerns are maintained independently. Therefore, the adaptability, reusability and maintainability of the service system will be increased.

In the future, a systematic method to efficiently identify PRAs and SRAs will be examined.

References

[1] Rania Khalaf, Nirmal Mukhi, Sanjiva Weerawarana, Service-oriented composition in BPEL4WS, in: WWW (Alternate Paper Tracks), 2003, pp. 27–28.
[2] Anis Charfi, Mira Mezini, Hybrid web service composition: business processes meet business rules, in: Proceedings of the 2nd International Conference on Service Oriented Computing. ACM, 2004, pp. 30–38.
[3] Maria Augustina Cibrán, Bart Verheecke, Dynamic business rules for web service composition, in: 2nd Dynamic Aspects Workshop (DAW05), 2005, pp. 13–18.

[4] María Agustina Cibrán, Maja D'hondt, Viviane Jonckers, Aspect-oriented programming for connecting business rules, in: Proceedings of the 6th International Conference on Business Information Systems, 2003, p. 24.
[5] Anis Charfi, Mira Mezini, Aspect-oriented web service composition with AO4BPEL, in: Web Services, Springer Berlin Heidelberg, 2004, pp. 168–182.
[6] Ivar Jacobson, Pan-Wei Ng, Aspect-oriented Software Development with Use Cases (Addison-wesley Object Technology Series), Addison-Wesley Professional, 2004.
[7] Gregor Kiczales, et al., Aspect-oriented programming, in: ECOOP'97—Object-oriented Programming, Springer Berlin Heidelberg, 1997, pp. 220–242.
[8] Adam Przybyłek, Where the truth lies: AOP and its impact on software modularity, in: Fundamental Approaches to Software Engineering, Springer Berlin Heidelberg, 2011, pp. 447–461.
[9] Shaukat Ali, Tao Yue, Lionel C. Briand, Does aspect-oriented modeling help improve the readability of UML state machines? Softw. Syst. Model. 13 (3) (2014) 1189–1221.
[10] Germán H. Alférez, et al., Dynamic adaptation of service compositions with variability models, J. Syst. Softw. 91 (2014) 24–47.
[11] Eric Bodden, Éric Tanter, Milton Inostroza, Join point interfaces for safe and flexible decoupling of aspects, ACM Trans. Softw. Eng. Methodol. (TOSEM) 23 (1) (2014) 7.
[12] W3C Web Services WG, Web Services Architecture, http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/.W3C, Working Group Note 11, Feb 2004.
[13] Roy W. Schulte, Yefim V. Natis, Service Oriented Architecture, Gartner Group, SSA Research Note SPA-401-068, 1996.
[14] Chankyu Park, et al., Knowledge-based AOP framework for business rule aspects in business process, ETRI J 29 (4) (August 2007).
[15] B2J, http://www.eclipse.org/stp/b2j.